Reference Issues, and Events

Ken Getz
keng@mcwtech.com
@kengetz

Avoiding Reference Problems

- When working with classes, easy to get into trouble
- Need to have a rudimentary understanding of what's going on
- This section points out an important, common problem

Object Variables: They're Pointers

- To use class module, create variable and instantiate

```
Dim objFile As TextFile
Set objFile = New TextFile
```

- Object variable is, internally, a pointer to newly created object

Holding Onto That Balloon

- **As long as *objFile* exists, holding onto the object…**
  - TextFile object exists, consuming resources
- **What if *objFile* goes out of scope?**
  - Or if you set it to Nothing?

```
Set objFile = Nothing
```

- **At that point, there is no reference to object**

Multiple Pointers

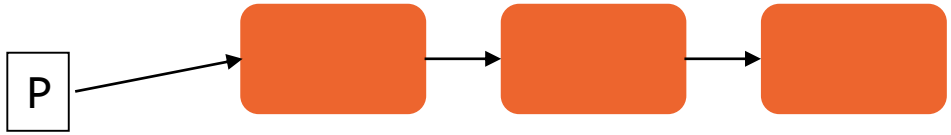
- **Possible to have more than one variable referring to the same object**
  - Set one object variable equal to another

```
Dim objFile2 As TextFile
Set objFile2 = objFile
```

- **Setting one variable to Nothing doesn't destroy the object**
  - Because there's still a reference to the object
  - Setting second variable to Nothing does destroy the object—no more references!

Moving On From Balloons

- **Balloons make a nice metaphor**
- **Often need self-referential classes**
- **Can be useful for abstract data structures**
  - Stacks
  - Linked Lists
  - Queues
  - Covered in separate module



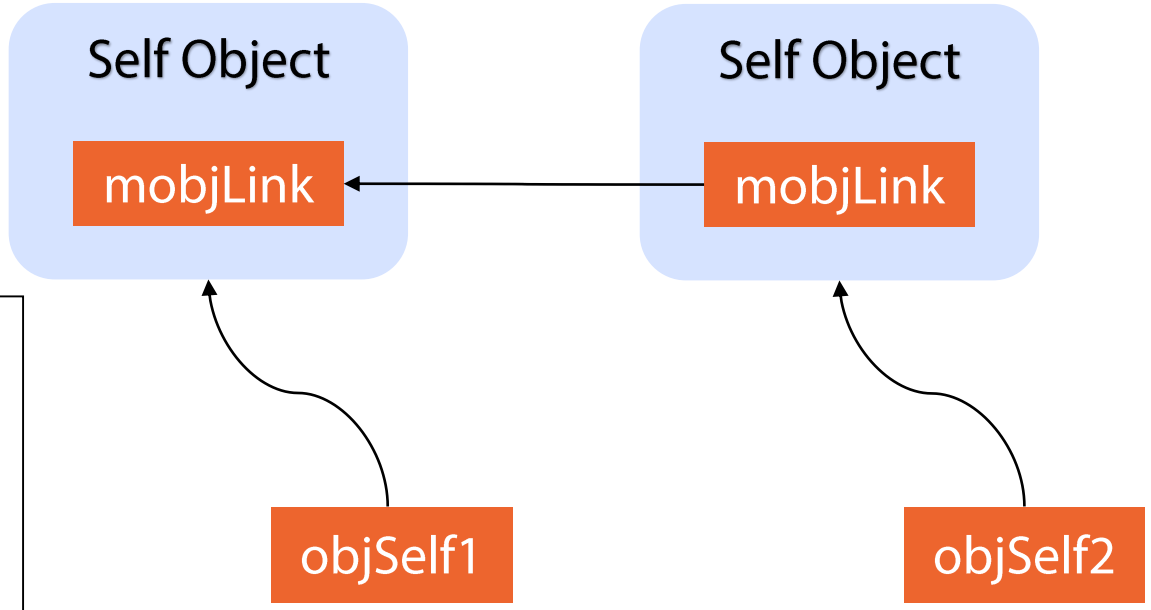
Termination Issues

- Check out Self class, which has a reference to another instance of the same class
- Sample code creates two instances of Self class
- Watch what happens!

DEMO 6

- Examine Self class
- Examine TestSelf1 procedure

What Just Happened?




```
Dim objSelf1 As Self
Dim objSelf2 As Self

Set objSelf1 = New Self
Set objSelf2 = New Self

Set objSelf2.Link = objSelf1

Set objSelf1 = Nothing
Set objSelf2 = Nothing
```

Not Elegant

- In this sample, both objects ended up terminated and removed from memory
- The real problem occurs when you have circular references

DEMO 7

- Create circular reference
- Examine SelfTest2 procedure

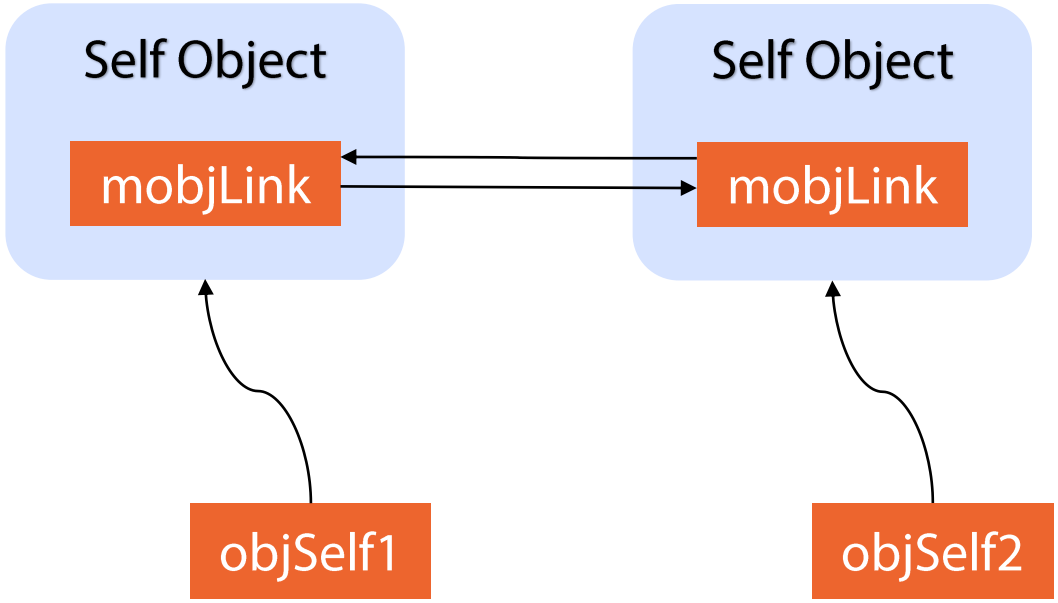
What Just Happened?

```
Dim objSelf1 As Self
Dim objSelf2 As Self

Set objSelf1 = New Self
Set objSelf2 = New Self

Set objSelf2.Link = objSelf1
Set objSelf1.Link = objSelf2

Set objSelf1 = Nothing
Set objSelf2 = Nothing
```



Leaving Objects In Memory

- Not a good thing!
- Do this too often, and you run out of memory
- To solve this problem, your code must clean up after itself

Cleaning Up

- **Solution is simple, but requires some effort**
- **You must:**
  - Add a method to your class that destroys all object references the class contains
  - Call the method prior to destroying a reference to an instance of the class
  - Explicitly destroy references to objects (set them to Nothing) rather than counting on VBA to do it for you

DEMO 8

- Investigate CleanUp method
- Investigate TestSelf3

What Just Happened?

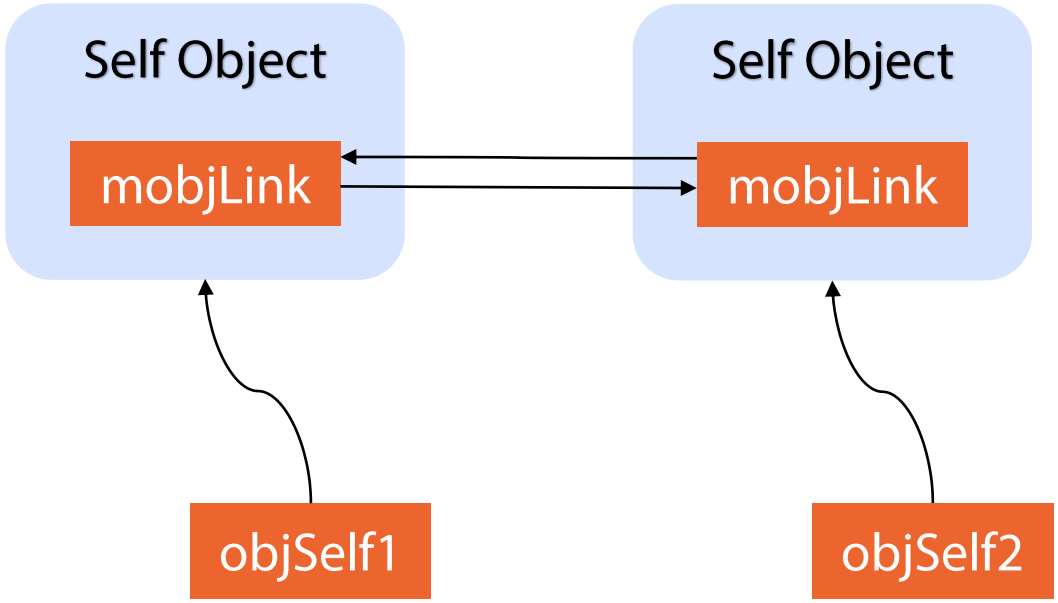
```
Dim objSelf1 As Self
Dim objSelf2 As Self

Set objSelf1 = New Self
Set objSelf2 = New Self

Set objSelf2.Link = objSelf1
Set objSelf1.Link = objSelf2

objSelf1.Cleanup
objSelf2.Cleanup

Set objSelf1 = Nothing
Set objSelf2 = Nothing
```



What to Take Away?

- **If a class includes a reference to another object…**
  - Make sure you include a procedure, like Cleanup
  - Make sure and call Cleanup before setting only reference to the object to Nothing
- **Avoid costly memory-hogging bugs**
  - Don't allow objects to become orphaned!

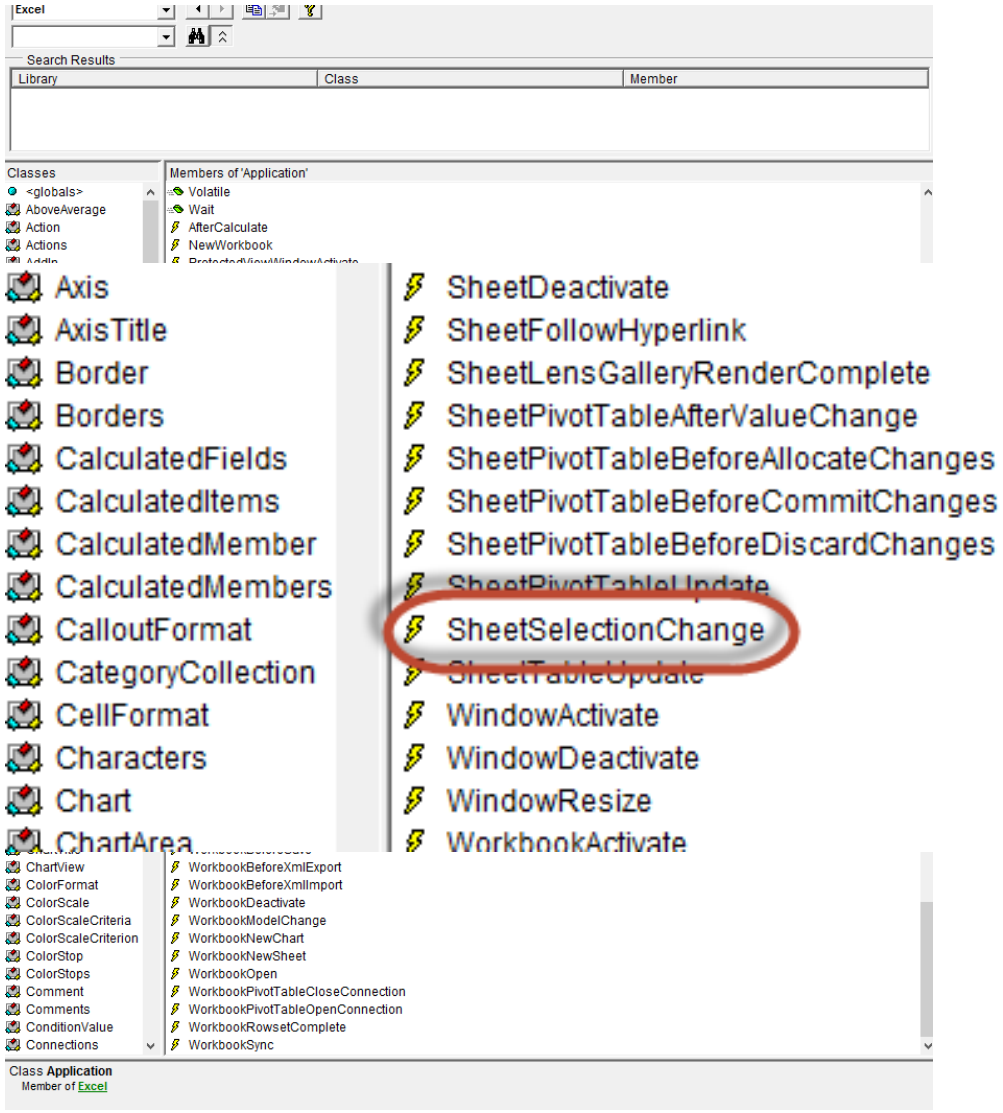
Tapping Into Events

- **No way to avoid handling events**
- **How does VBA know about events?**
  - At load time, queries objects' type libraries
  - Registers event sinks for each event that has a handler
- **When event is triggered, runs associated VBA code in event sink**
- **Can create your own events, and handle them**
  - **WithEvents** keyword allows classes to handle events
  - **Event** declaration and **RaiseEvent** keyword allow classes to raise events

What is WithEvents?

- VBA keyword
- Used in conjunction with object variable declaration
- Signals to VBA that you want VBA to notify you of any events the object exposes
- Check Object Browser for information

Excel Application Events



Handling Events

- **Code to handle event must be in a class module**
- **Declaration for WithEvents variable must be in Declaration section**
  - At the top of the class module
  - Can't be in the body of a procedure
  - Must be a "class-level" variable

Using WithEvents

- **Declare variable that refers to object raising events:**

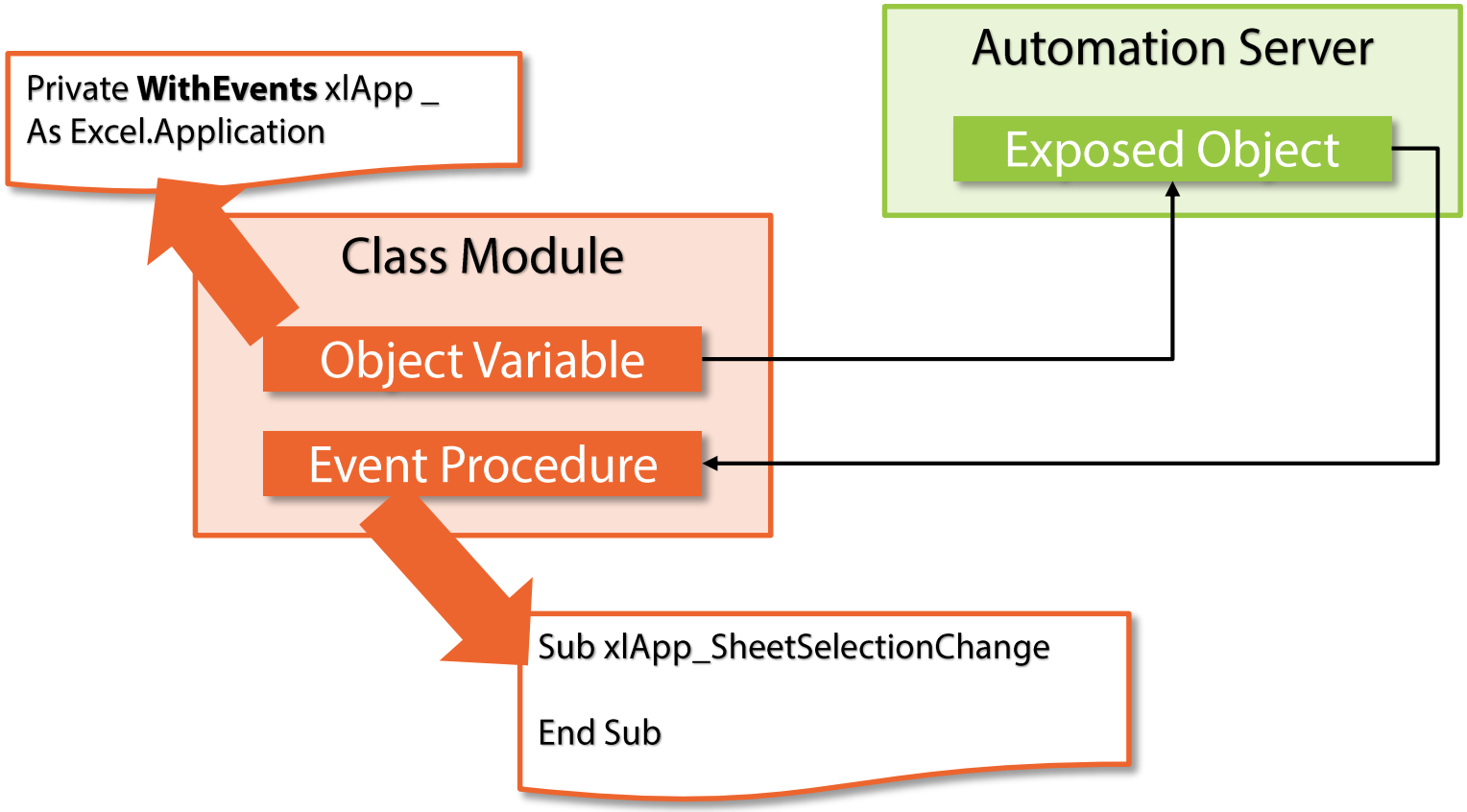
```
Private WithEvents xlApp as Excel.Application
```

- **Sample loads instance of Excel, and gathers event information back from the Excel Application object**

DEMO 9

- Create demo form

What Just Happened?



Events in Class Modules

- **Form automatically loads its class module when form loaded**
  - If WithEvents in standard class module…
  - Must create an instance in order to handle events
- **Why only class modules?**
  - Standard modules have no means of handling events
  - Only class modules include this functionality
  - Can be standalone class module, or any code-behind class module
  - Simply add WithEvents keyword to an object declaration
    - Object instance can handle events

Warning!

- Watch out!
- Event procedures created using WithEvents are simply functions that VBA calls in reaction to an event
- Event raiser can't go on until event handler completes
- Don't put MsgBox in event handler

DEMO 10

- Add MsgBox in event handler

Raising Events

- **Want to allow your own classes to notify listeners when something occurs within an object instance?**
  - Raise events
- **Any class that creates an instance of your class can be notified when something interesting happens in your object**
  - Just need to raise events from your class
  - Much like the way the Excel Application object works

Adding Events to Your Classes

- **Declare the events in the class module's Declarations area**

```
' Event declarations
Public Event ReadLine(ByVal Text As String)
Public Event WriteLine(Text As String, Skip As Boolean)
Public Event AfterOpen()
Public Event BeforeClose(Cancel As Boolean)
```

- **Add RaiseEvent statement in code when time to raise an event**

```
Do Until EOF(mhFile)
    Line Input #mhFile, strLine

    ' Raise ReadLine event
    RaiseEvent ReadLine(strLine)

    Me.TextLines.Add strLine
Loop
```

- **RaiseEvent makes method call to each listener**
  - Calls event handler in every object that handles the event

DEMO 11

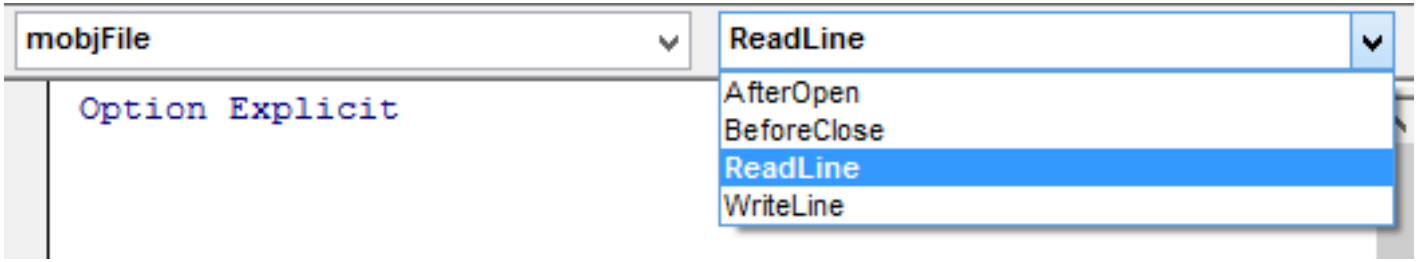
- Investigate TextFile3 class
- Check out events

Sinking Events

- **To react to events raised by an object**
  - Must use WithEvents keyword
  - Same for custom objects as it is for built-in objects
- **In class module, add reference to object raising events**

```
Private WithEvents mobjFile As TextFile3
```

- **Once variable added, look in Object drop-down list**
- **Look in Procedures list for events to sink**



- **Create object variable instance**

DEMO 12

- Create instance of TextFile3 class and handle events

Summary

- **Must keep track of references when using object properties**
  - Provide some means of cleaning up outgoing references before destroying an object
  - Easy to end up with dangling references and objects that won't "die"
- **Easy to handle events in VBA: Use the WithEvents keyword**
  - Easy to add events to your own objects, using the Event declaration and RaiseEvent keyword
- **Be careful not to block event handler**